CCA
Common Component Architecture

The CCA ISIC and You (aka CCTTSS)

PI: Rob Armstrong
rob@sandia.gov

Co-Investigators:
David Bernholdt (ORNL), Lori Freitag Diachin (SNL), Dennis Gannon (Indiana Univ.), James Kohl (ORNL), Gary Kumfert (LLNL), Lois Curfman McInnes (ANL), Jarek Nieplocha (PNNL), Steven Parker (Univ. of Utah), Craig Rasmussen (LANL)

http://www.cca-forum.org/ccttss

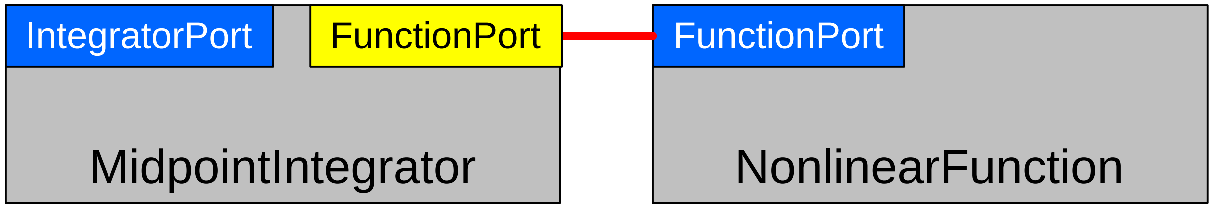
March 2003 PI Meeting

Outline

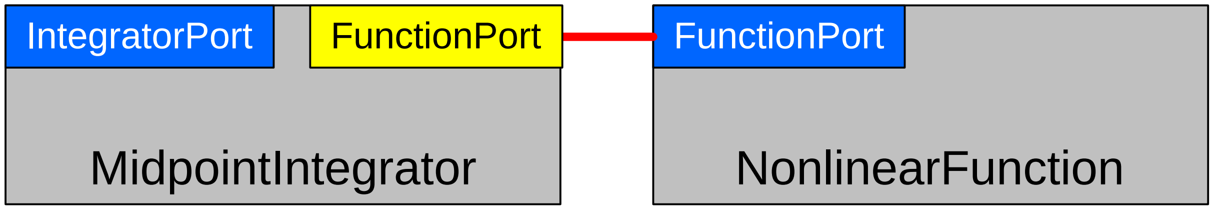
- CCA Components: what are they?
- Working with the CCA
  - Software
  - Tutorials
  - Getting help: CCA adepts and community
- Future of the CCA

What *are* components, frameworks?

- **No universally accepted definition…yet**
  - My working def’n: software that is composable
  - Framework is everything that isn’t a component
- **Interacts with the outside world *only* through well-defined interfaces**
  - Implementation is opaque to the outside world
- **Can be composed with other components**
  - “Plug and play” model to build applications
  - Composition based on these interfaces



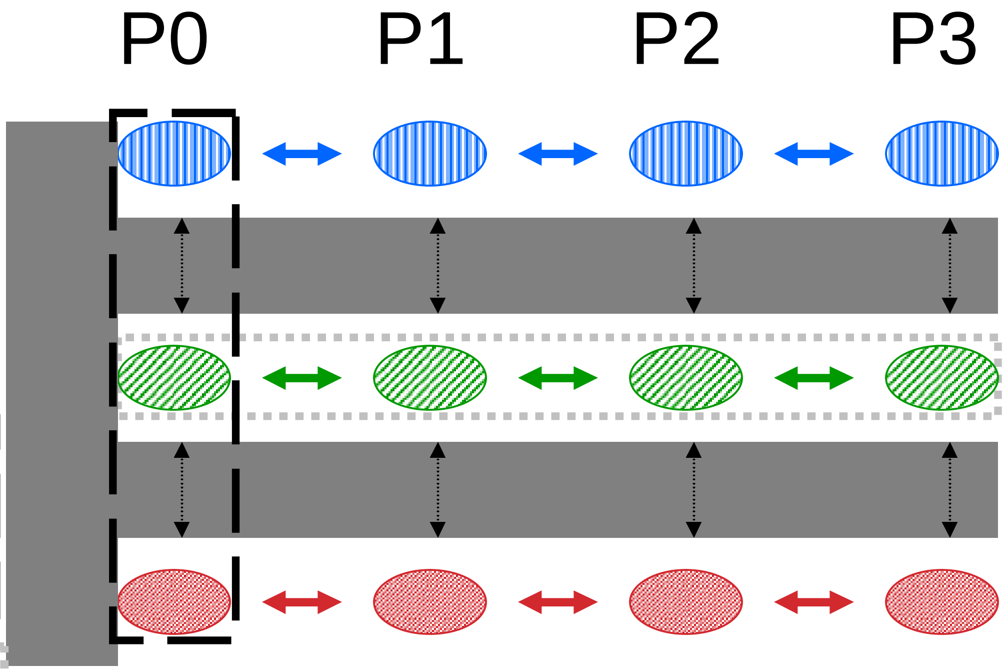
Ports: Connections Interface Exchange



- Components interact through well-defined interfaces, or *ports*
  - In OO languages, a port is a virtual class or interface
  - In Fortran, a port is a bunch of subroutines or a module
- Components may *provide* ports – implement the class or subroutines of the port
- Components may *use* ports – call methods or subroutines in the port
- Links denote a caller/callee relationship, ***not dataflow!***
  - e.g., FunctionPort could contain: *evaluate(in Arg, out Result)*

Framework Stays "Out of the Way" of Component Parallelism

- Single component multiple data (SCMD) model is component analog of widely used SPMD model
- Each process loaded with the same set of components wired the same way
- Different components in same process "talk to each" other via ports and the framework
- ***Same component in different processes talk to each other through their favorite communications layer (i.e. MPI, PVM, GA)***



Components: Blue, Green, Red

Framework: Gray

MCMD/MPMD also supported

Working with CCA

- The CCA components are easy to build
  - Designed to easily import existing code into components
  - Reversible – you don’t “lose” the original code
- Significant components available to download
  - Buildable code right now
  - RPMs real soon now
- Tutorials are hands on and no nonsense
  - You *will* be able to build a component
  - Tutorials are usually conducted in and around CCA mtg’s.
  - Copies of all tutorial presentations are on the web

Making your code CCA compliant in 2 easy steps

1. Create a single method, function,or subroutine in C++, C, or Fortran
   - C++ SIDL rendering:
     `void setServices(Services svc) { /*...*/}`
2. There isn’t a “2”.

- *This won’t make a very interesting component.*

More interesting component:
put more code here

CCA leaves your code alone

- Everything that makes your component a component is right here:

  `void setServices(Services svc) { /*...*/}`

- The rest of the code is yours.
- The really hard part we have yet to talk about: carefully define what functionality your component
  - *Provides* to the outside world
  - *Uses* from the outside world

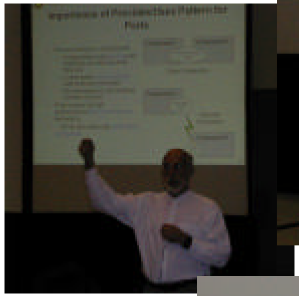
CCA Software: Component Inventory

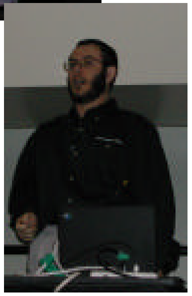
- **Services and Performance**
  - ***Ccaffeine*** – SNL – Services for parameter ports, connections between SIDL and classic ports, MPI access, etc.
  - ***Performance Component*** – Oregon/LANL/SNL – classic & SIDL – Measurement capabiliites for components
- **Data Management, Meshing, and Discretization**
  - ***Global Array Component*** – PNNL – classic & SIDL – Multidimensonal dense distributed arrays
  - ***TSTTMesh*** – ANL/SNL – classic – Unstructured mesh management
  - ***FEMDiscretization*** – ANL/SNL – classic – Finite element discretization
  - ***GrACEComponent*** – SNL – classic – Parallel structured adaptive mesh refinement
- **Integration and Optimization**
  - ***CvodeComponent*** – LLNL (TOPS) – classic – Implicit ODE integrators
  - ***TAOSolver*** – ANL – SIDL – Solvers for constrained and unconstrained optimization
- **Parallel Data Description, Redistribution, and Visualization**
  - ***DistArrayDescriptorFactory*** – ORNL – classic – Uniform means for describing dense multi-dimensional arrays
  - ***CumulvsMxN*** – ORNL – classic – “MxN” parallel data redistribution
  - ***VizProxy*** – ORNL – classic – Companion “MxN” endpoint for passing data to front-end viewers for graphical rendering
- **Scientific Applications**
  - **Chemistry components** – SIDL – PNNL/SNL – electronic structure components based on NWChem and MPQC
  - **Combustion components** – classic – SNL – based on GrACEComponent and physical/chemical models

RPM coordinator: B. Allan (SNL)

The CCA Tutorial

- CCA Forum Tutorial Working Group
  - Rob Armstrong (SNL), David Bernholdt (ORNL, chair), Lori Freitag Diachin (SNL), Wael Elwasif (ORNL), Dan Katz (JPL), Jim Kohl (ORNL), Gary Kumfert (LLNL), Lois Curfman McInnes (ANL), Boyana Norris (ANL), Craig Rasmussen (LANL), Jaideep Ray (SNL), Torsten Wilde (ORNL)
- Highlights:
  - CCA Concepts
  - A Simple Component Example
  - Language Interoperability using Babel
  - Writing CCA Components
  - More Complex Component-Based Applications
  - CCA Status and Plans





Tutorial Presentations

Jan 2002	CCA Forum Mtg	Santa Fe, NM
Apr 2002	CCA Forum Mtg	Townsend, TN
Jun 2002	CCA Forum Mtg	ANL
Sep 2002	ACTS Collection Workshop	LBL
Oct 2002	CCA Forum Mtg	Half Moon Bay, CA
Oct 2002	LACSI Symposium	Santa Fe, NM
Nov 2002	SC2002	Baltimore, MD
Jan 2003	CCA Forum Mtg	Pasadena, CA

Getting help with CCA technology

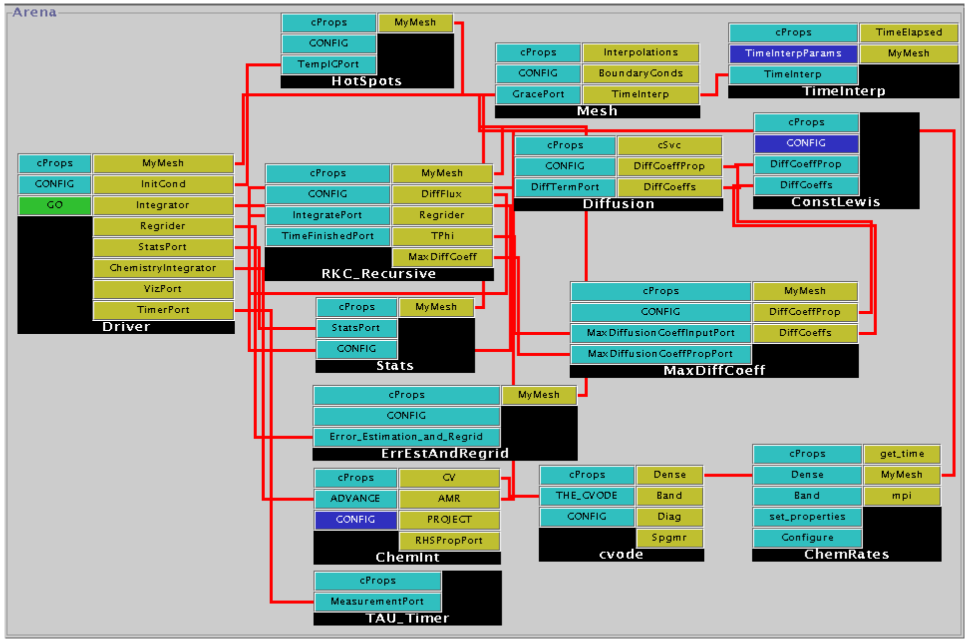
- Write to: *cca-pi@cca-forum.org*
- Usually adopters of CCA have a CCTTSS liaison assigned to them
- Online manuals, tutorials:

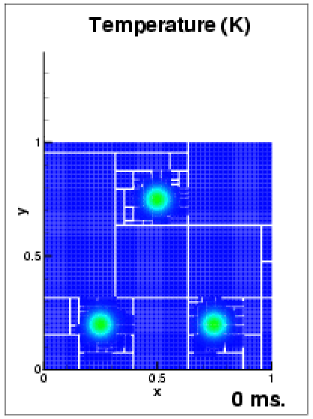
  http://www.cca-forum.org

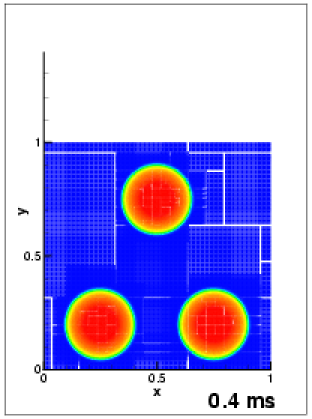
CCA
Common Component Architecture

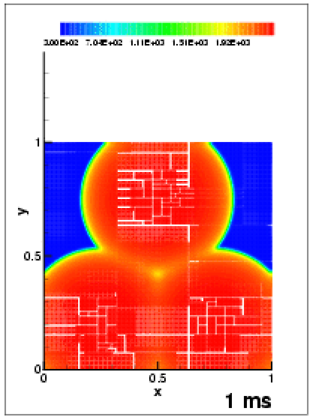
Reacting Flow Software “Facility”

- *A Computational Facility for Reacting Flow Science* (CFRFS), SciDAC application center, PI: H. Najm
- Epitome of componentized applications
  - There is no one application, but a component “facility”


Arena
HotSpots: cProps, MyMesh, CONFIG, TempICPort
Mesh: cProps, Interpolations, CONFIG, BoundaryConds, GracePort, TimeInterp
TimeInterp: cProps, TimeElapsed, TimeInterpParams, MyMesh, TimeInterp
Driver: cProps, MyMesh, CONFIG, InitCond, GO, Integrator, Regrider, StatsPort, ChemistryIntegrator, VizPort, TimerPort
RKC_Recursive: cProps, MyMesh, CONFIG, DiffFlux, IntegratePort, Regrider, TimeFinishedPort, TPhi, MaxDiffCoeff
Diffusion: cProps, cSvc, CONFIG, DiffCoeffProp, DiffTermPort, DiffCoeffs
ConstLewis: cProps, CONFIG, DiffCoeffProp, DiffCoeffs
Stats: cProps, MyMesh, StatsPort, CONFIG
MaxDiffCoeff: cProps, MyMesh, CONFIG, DiffCoeffProp, MaxDiffusionCoeffInputPort, DiffCoeffs, MaxDiffusionCoeffPropPort
ErrEstAndRegrid: cProps, MyMesh, CONFIG, Error_Estimation_and_Regrid
ChemInt: cProps, CV, ADVANCE, AMR, CONFIG, PROJECT, RHSPropPort
cvode: cProps, Dense, THE_CVODE, Band, CONFIG, Diag, Spgmr
ChemRates: cProps, get_time, Dense, MyMesh, Band, mpi, set_properties, Configure
TAU_Timer: cProps, MeasurementPort







Future of CCA

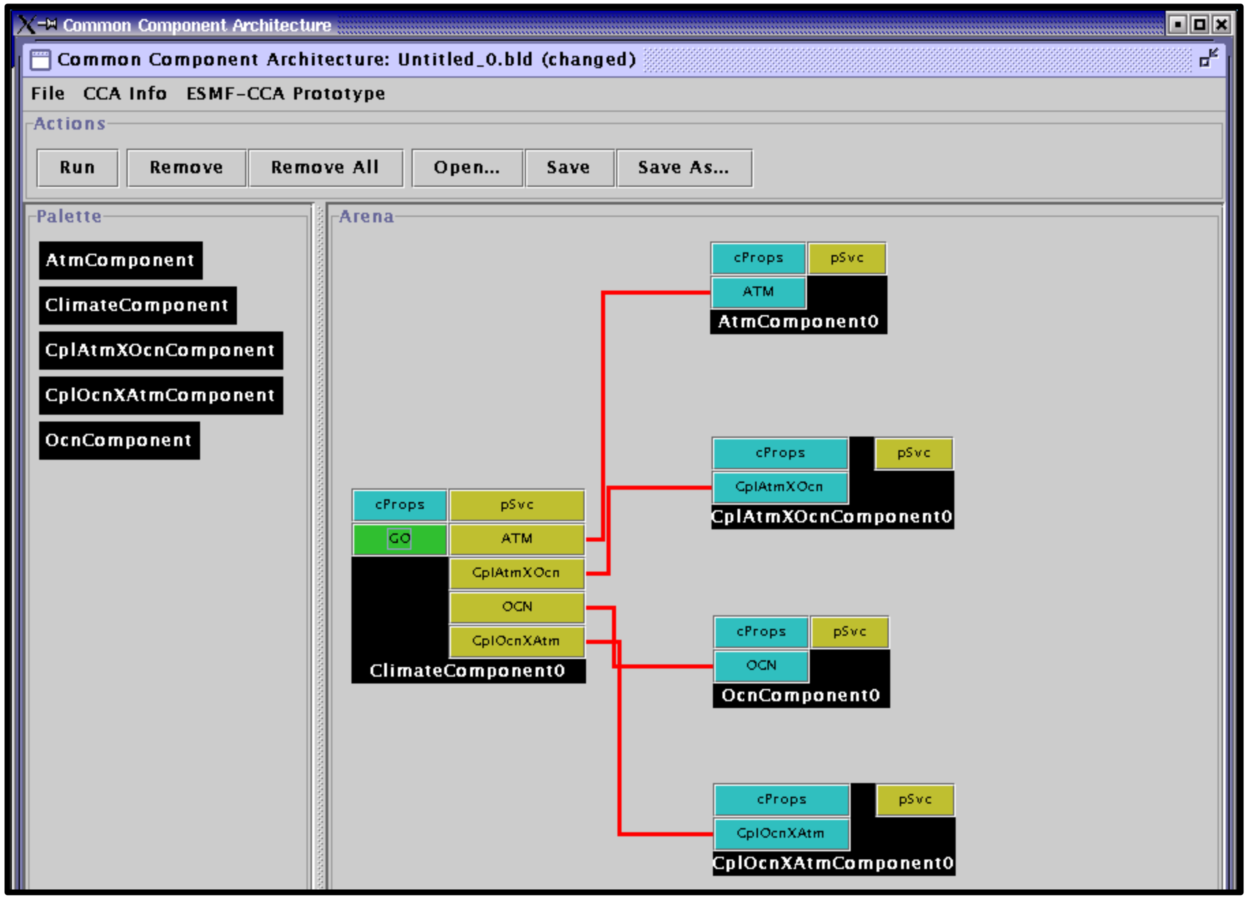
- Current big adopters are:
  - Combustion (CFRFS)
  - Quantum Chemistry
- This year
  - big climate apps push
  - a couple approaches for Fortran components exist
    - climate and fusion early adopters
- More and better components
  - structured, unstructured meshes (APDEC, TSTT)
  - MxN, parallel IO
  - refinements of existing frameworks and components

CCA will help you componentize your code

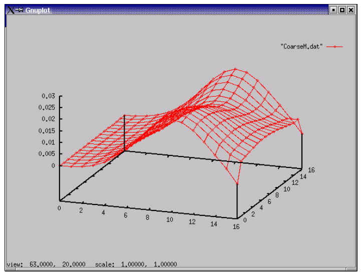
- Converting your code is easy and reversible
  - Creating interfaces is the hard part
- Significant and useful software in the form of frameworks and components are available
- Much documentation and tutorials available

CCA Impact Outside of SciDAC

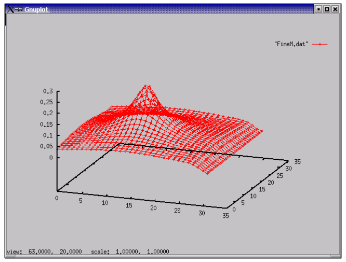
- Climate prototype (Courtesy Shujia Zhou, ESMF)



ATM



OCN


Gnuplot
"FineM.dat"